Elements Of Statistical Learning Notes

1 Introduction

Here are some examples of learning problems: heart attack prediction, stock price prediction, handwritten number identification, serum glucose estimation, prostate cancer risk factor identification. Supervised learning has outcome variable in data to guide the learning process. Unsupervised learning is to describe how the data are organised or clustered.

2 Overview Of Supervised Learning

2.1 Introduction

Supervised learning. Use the inputs to predict the values of the outputs. Predictors/features/independent variables to predict responses/dependent variables.

2.2 Variable Types And Terminology

Qualitative variables are known as categorical or discrete. Regression to predict quantitative and classification to predict qualitative. Both can be viewed as a task in function approximation. Ordered categorical like small, medium, large buckets. Dummy variables are the proper name for one hot encodings for K bins buckets categories a binary vector of size K with 1 of 1 and $K-1$ of 0s is a symmetric representational scheme.

2.3 Two Simple Approaches To Prediction: Least Squares And Nearest Neighbours

Recall the linear model. Predict the output:

$$\hat{Y} = \hat{\beta}_0 + \sum X_j \hat{\beta}_j$$

Where $\hat{\beta}_0$ is the intercept. So for example one can instead simply choose to include 1 in the inputs vector X and obtain like:

$$\hat{Y} = X^T \hat{\beta}$$

There exist other metrics but by far the most popular method for fitting a linear model to a set of training data is to have β minimise the residual sum of squares:

$$\text{RSS}(\beta) = \sum (y_i - x_i^T \beta)^2$$

$\text{RSS}(\beta)$ is a quadratic function of the parameters.

$$\text{RSS}(\beta) = (y - X\beta)^T (y - X\beta)$$
$$X^T (y - X\beta) = 0$$
$$\hat{\beta} = (X^T X)^{-1} X^T y$$

A bunch of blue and orange points in $\mathbb{R}^2$ and a line linear binary predictor model e.g. an inequality where dudes are predicted based on side however they then present what appears to be a strongly outperforming model based upon the nearest-neighbour methods. In this case:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

Where $N_k(x)$ is the neighbourhood of x defined by the k closest points x_i in the training sample under here the Euclidean distance metric. A simple majority casting in to the binary variable e.g. each dude is predicted to be whatever colour the majority of its 15-nearest-neighbours in the training data were. This produces a boundary curve which might be plausible for some data sets. The 1-nearest-neighbour scheme which produces a Voronoi tessellation of polygons from implicit perpendicular bisectors bounding nearest regions. This produces a totally garbage overfitting regions decision boundary.

Randomly generated training and test data to make a point about these models and it produces a graph suggesting that for 200 such generated points the model minimising test error may be a 9-nearest-neighbour model.

A large subset of the most popular techniques in use today are variants of these two simple procedures. Kernel methods use weights which smoothly decrease to 0 based on distance from the target point, rather than the simple binary $0, 1$ weights induced by k-nearest-neighbours. So these distance kernels metrics can be modified to emphasise some variable more than others and one supposes there exist natural such families. Local regression fits linear models by locally weighted least squares, rather than fitting constants locally. Projection pursuit and neural network models consist of sums of non linearly transformed linear models.

2.4 Statistical Decision Theory

Formalised squared error loss function example conditioning on X the conditional expectation, also known as the regression function. Loss functions and a Bayes optimal classifier decision boundary graphic. The error rate of the Bayes classifier is called the Bayes rate. The relatively easy to compute k-nearest-neighbours model can be a good approximation though precise details are left vague.

2.5 Local Methods In High Dimensions

The stable but biased linear model and the less stable but apparently less biased class of k-nearest-neighbours estimates. With a reasonably large set of training data one would think nearest-neighbours would be good but this approach breaks down in high dimensions. Sparse sampling and that in high dimensions most sample points are close to a boundary of the sample.

2.6 Statistical Models, Supervised Learning

Our goal is to find a useful approximation $\hat{f}(x)$ to the function $f(x)$ that underlies the predictive relationship between the inputs and outputs. In the theoretical setting of

Section 2.4 we saw that squared error loss led us to the regression function $f(x) = E(Y|X = x)$ for a quantitative response.

Key is that treating supervised learning as a problem in function approximation encourages the geometrical concepts of Euclidean spaces and mathematical concepts of probabilistic inference to be applied to the problem. Linear basis expansion. Transformations of the input vector x exist like polynomial and trigonometric functions so h_k might be $x_1^2, x_1x_2^2, \cos(x_1)$ and so on. We also encounter nonlinear expansions, such as the sigmoid transformation common to neural network models,

$h_k(x) = \frac{1}{1+e^{-x^T\beta_k}}$

Recall maximum likelihood estimation. The principle assumes that the most reasonable values for θ are those for which the probability of the observed sample is maximised. Least squares for the additive error model $Y = f_\theta(X) + \epsilon$, with $\epsilon \approx N(0, \sigma^2)$, is equivalent to maximum likelihood using the conditional likelihood.

$\Pr(Y|X, \theta) = N(f_\theta(X), \sigma^2)$

Note that log-likelihood is also known as cross-entropy.

2.7 Structured Regression Models

In general the constraints imposed by most learning methods can be described as complexity restrictions of one kind or another. This usually means some kind of regular behaviour in small neighbourhoods of the input space. That is, for all input points x sufficiently close to each other in some metric, $\hat{f}$ exhibits some special structure such as nearly constant, linear, or low-order polynomial behaviour. The estimator is then obtained by averaging or polynomial fitting in that neighbourhood.

2.8 Classes Of Restricted Estimators

Roughness penalty and Bayesian methods. Kernel methods and local regression. For example the Gaussian kernel has a weight function based on the Gaussian density function and assigns weights to points that die exponentially with their squared Euclidean distance from x_0. Bias functions and dictionary methods.

2.9 Model Selection And The Bias-Variance Tradeoff

A smoothing or complexity parameter that has to be determined: the multiplier of the penalty term, the width of the kernel, or the number of basis functions. The expected prediction error is known as test or generalisation error. They kind of insinuate a middle ground of bias variance, and prediction error minimisation on model complexity. Bias-variance tradeoff. So we will study some things about model complexity and various types of tasks as well as

perhaps how one goes about eyeballing data sets and selecting models based upon observations.

3 Linear Methods For Regression

3.1 Introduction

A linear regression model assumes that the regression function is linear in the input. They are simple and often provide an adequate and interpretable description of how the inputs affect the output. For prediction purposes they can sometimes outperform fancier nonlinear models, especially in situations with small numbers of training cases, low signal-to-noise ratio or sparse data. Finally, linear methods can be applied to transformations of the inputs and this considerably expands their scope. These generalisations are sometimes called basis-function methods.

3.2 Linear Regression Models And Least Squares

Linear model assumes either the regression function $E(Y|X)$ is linear, or that the linear model is a reasonable approximation. The variables X_j can come from different sources: quantitative inputs, transformations of quantitative inputs, such as log, square-root, or square, basis expansions, such as $X_2 = X_1^2, X_3 = X_1^3$, leading to a polynomial representation, dummy encoding of the levels of qualitative inputs, interactions between variables, for example $X_3 = X_1 \cdot X_2$.

In their prostate cancer example they just kind of took the log of a bunch of things and showed a kind of R style output of running a multiple linear regression. Then they talk about the meanings of coefficient, std. error, and Z score, as well as the F statistic notion on model simplicity. And consider dropping non significant predictors and re generating a new linear model as I have executed in the R language before multiple times.

The Gauss Markov Theorem asserts that the least squares estimates of the parameters β have the smallest variance among all linear unbiased estimators.

However, there may well exist a biased estimator with smaller mean squared error. Such an estimator would trade a little bias for a larger reduction in variance. Biased estimators are commonly used.

Regression by successive orthogonalisation e.g. Gram-Schmidt algorithm. The multiple regression coefficient $\hat{\beta}_j$ represents the additional contribution of x_j on y, after x_j has been adjusted for all the other x_i.

3.3 Subset Selection

There are 2 reasons why we are often not satisfied with the least squares estimates. The first is prediction accuracy: the least squares estimates

often have low bias but large variance. The second reason is interpretation: with a large number of predictors we often would like to determine a smaller subset that exhibit the strongest effects.

Best subset regression finds for each $k \in \{0, 1, 2, \ldots, p\}$ the subset of size k that gives the smallest residual sum of squares. An efficient algorithm known as the leaps and bounds procedure makes this feasible for $p \leq 30, 40$. Note critically that for example the best subset of size 2 need not include the 1 variable that was the best subset of size 1. The question of how to choose k involves the tradeoff between bias and variance, along with the more subjective desire for simplicity. There are a number of criteria that one may use; typically we choose the smallest model that minimises an estimate of the expected prediction error.

Cross-validation to estimate prediction error and select k is one technique. The Akaike Information Criterion is a popular alternative.

For $p >> 40$ for example we can greedily seek a good path through them. Forward-stepwise selection starts with the intercept, and then sequentially adds in to the model the predictor that most improves the fit. With many predictors, clever updating algorithms can exploit the orthogonal upper decomposition for the current fit to rapidly establish the next candidate. Perhaps this performs well enough for some use cases.

Backward-stepwise selection starts with the full model, and sequentially deletes the predictor that has the least impact on the fit. The candidate for dropping is the variable with the smallest Z-score. Backward selection can only be used when $N > p$.

Some software packages implement hybrid stepwise-selection strategies that consider both forward and backward moves at each step, and select the "best" of the two. For example in the R package the step function uses the Akaike Information Criterion for weighing the choices, which takes proper account of the number of parameters fit; at each step an add or drop will be performed that minimises the Akaike Information Criterion score.

Finally, we note that often variables come in groups (such as the dummy variables that code a multi-level categorical predictor). Smart stepwise procedures (such as step in R) will add or drop whole groups at a time, taking proper account of their degrees-of-freedom.

Forward-stagewise regression is even more constrained than forward stepwise regression. It starts like forward-stepwise regression, with an intercept equal to $\hat{y}$, and centered predictors with coefficients initially all 0. At each step the algorithm identifies

the variable most correlated with the current residual. It then computes the simple linear regression coefficient of the residual on this chosen variable, and then adds it to the current coefficient for that variable. This is continued till none of the variables have correlation with the residuals - i.e. the least-squares fit when $N > p$. Historically has been dismissed as being inefficient. It turns out that this "slow fitting" can pay dividends in high-dimensional problems.

3.4 Shrinkage Methods

By retaining a subset of the predictors and discarding the rest, subset selection produces a model that is interpretable and has possibly lower prediction error than the full model. However, because it is a discrete process - variables are either retained or discarded - it often exhibits high variance, and so doesn't reduce the prediction error of the full model. Shrinkage methods are more continuous, and don't suffer as much from high variability.

L2 Ridge regression shrinks the regression coefficients by imposing a penalty on their size. The L2 Ridge coefficients minimise a penalised residual sum of squares. The idea of penalising by the sum-of-squares of the parameters is also used in neural networks, where it is known as weight decay. L2 Ridge regression can also be derived as the mean or mode of a posterior distribution, with a suitably chosen prior distribution.

The largest principal component is the direction that maximises the variance of the projected data, and the smallest principal component minimises that variance. L2 Ridge regression projects y onto these components, and then shrinks the coefficients of the low-variance components more than the high-variance components.

The implicit assumption is that the response will tend to vary most in the directions of high variance of the inputs. This is often a reasonable assumption, since predictors are often chosen for study because they vary with the response variable, but need not hold in general.

The L1 Lasso is a shrinkage method like L2 Ridge, with subtle but important differences. The L1 Lasso estimate is defined by basically a simple linear L_1 sum of magnitudes constraint function on normed coefficients. In the signal processing literature, the L1 Lasso is also known as basis pursuit. Computing the L1 Lasso solution is a quadratic programming problem, although efficient algorithms are available for computing the entire path of solutions as λ is varied, with the same computational cost as for L2 Ridge regression.

However, the nature of the shrinkage is not obvious, and we investigate it further below. Like the subset size in

variable subset selection, or the penalty parameter in L2 Ridge regression, t should be adaptively chosen to minimise an estimate of expected prediction error.

And they further depict elliptical threshold least squares error function curves and assert that these algorithms produce the smallest such and the point where there exists an intersection with a sum of coordinates taxicab hypercube or hypersphere.

In this view, the L1 Lasso, L2 Ridge regression and best subset selection are Bayes estimates with different priors. Note, however, that they are derived as posterior modes, that is, maximisers of the posterior. It is more common to use the mean of the posterior as the Bayes estimate. L2 Ridge regression is also the posterior mean, but the L1 Lasso and best subset selection are not.

Least angle regression can be viewed as a version of forward stepwise regression. Least angle regression is intimately connected with the L1 Lasso, and in fact provides an extremely efficient algorithm for computing the entire L1 Lasso path. Least angle regression enters "as much" of a predictor as it deserves. At the first step it identifies the variable most correlated with the response. Rather than fit this variable completely, least angle regression moves the coefficient of this variable continuously toward its least squares value (causing its correlation with the evolving residual to decrease in absolute value). As soon as another variable "catches up" in terms of correlation with the residual, the process is paused. The second variable then joins the active set, and their coefficients are moved together in a way that keeps their correlations tied and decreasing. This process is continued until all the variables are in the model, and ends at the full least-squares fit.

Consider first a linear regression using a subset of k features. If this subset is pre specified in advance without reference to the training data, then the degrees of freedom used in the fitted model is defined to be k. Indeed, in classical statistics, the number of linearly independent parameters is what is meant by "degrees of freedom." Alternatively, suppose that we carry out a best subset selection to determine the "optimal" set of k predictors. Then the resulting model has k parameters, but in some sense we have used up more than k degrees of freedom.

3.5 Methods Using Derived Input Directions

In many situations we have a large number of inputs, often very correlated. The methods in this section produce a small number of linear combinations Z_m, $m = 1, \ldots, M$ of the original inputs X_j, and the Z_m are then used in place of the X_j as inputs in the regression.

The methods differ in how the linear combinations are constructed.

Principal Components Regression. In this approach the linear combinations Z_m used are the principal components as defined above. As with L2 Ridge regression, principal components depend on the scaling of the inputs, so typically we first standardise them. Note that if $M = p$, we would just get back the usual least squares estimates, since the columns of $Z = UD$ span the column space of X. For $M < p$ we get a reduced regression. We see that principal components regression is very similar to L2 Ridge regression: both operate via the principal components of the input matrix. L2 Ridge regression shrinks the coefficients of the principal components, shrinking more depending on the size of the corresponding eigenvalue; principal components regression discards the $p - M$ smallest eigenvalue components.

Partial Least Squares. This technique also constructs a set of linear combinations of the inputs for regression, but unlike principal components regression it uses y (in addition to X) for this construction. Like principal component regression, partial least squares is not scale invariant, so we assume that each x_j is standardised to have mean 0 and variance 1. Partial least squares begins by computing $\hat{\phi}_{1j} = (x_j, y)$ for each j. From this we construct the derived input $z_1 = \sum_j \hat{\phi}_{1j} x_j$, which is the first partial least squares direction. Hence in the construction of each z_m, the inputs are weighted by the strength of their univariate effect on y[3]. The outcome y is regressed on z_1 giving coefficient $\hat{\theta}_1$, and then we orthogonalise $x_1, \ldots, x_p$ with respect to z_1. We continue this process, until $M \leq p$ directions have been obtained. In this manner, partial least squares produces a sequence of derived, orthogonal inputs or directions $z_1, z_2, \ldots, z_M$. As with principal component regression, if we were to construct all $M = p$ directions, we would get back a solution equivalent to the usual least squares estimates; using $M < p$ directions produces a reduced regression.

3.6 Discussion: A Comparison Of The Selection And Shrinkage Methods

To summarise, partial least squares and L2 Ridge regression tend to behave similarly. L2 Ridge regression may be preferred because it shrinks smoothly, rather than in discrete steps. L1 Lasso falls somewhere between L2 Ridge regression and best subset regression, and enjoys some of the properties of each.

3.7 Multiple Outcome Shrinkage And Selection

To apply selection and shrinkage methods in the multiple output case, one could apply a univariate technique

individually to each outcome or simultaneously to all outcomes. With L2 Ridge regression, for example, we could apply formula (3.44) to each of the K columns of the outcome matrix Y, using possibly different parameters λ, or apply it to all columns using the same value of λ. The former strategy would allow different amounts of regularisation to be applied to different outcomes but require estimation of k separate regularisation parameters $\lambda_1, \ldots, \lambda_k$, while the latter would permit all k outputs to be used in estimating the sole regularisation parameter λ.

Other more sophisticated shrinkage and selection strategies that exploit correlations in the different responses can be helpful in the multiple output case. Combining responses is at the heart of canonical correlation analysis, a data reduction technique developed for the multiple output case. Similar to principal component analysis, canonical correlation analysis finds a sequence of uncorrelated linear combinations Xv_m, $m = 1, \ldots, M$ of the x_j, and a corresponding sequence of uncorrelated linear combinations Yu_m of the responses y_k, such that the correlations $\text{Corr}^2(Yu_m, Xv_m)$ are successively maximised.

3.8 More On The L1 Lasso And Related Path Algorithms

There exist some other academic notes and variants which are perhaps more performant. Incremental Forward Stagewise Regression, Piecewise-Linear Path Algorithms, The Dantzig Selector (wow, Emmanuel Candes and Terence Tao himself), The Grouped L1 Lasso.

A number of authors have studied the ability of the L1 Lasso and related procedures to recover the correct model, as N and p grow.

Pathwise Coordinate Optimisation. An alternate approach to the least angle regression algorithm for computing the L1 Lasso solution is simple coordinate descent. The idea is to fix the penalty parameter λ in the Lagrangian form and optimise successively over each parameter, holding the other parameters fixed at their current values.

3.9 Computational Considerations

Least squares fitting is usually done via the Cholesky decomposition of the matrix $X^T X$ or an orthogonal upper decomposition of X. With N observations and p features, the Cholesky decomposition requires $p^3 + N\frac{p^2}{2}$ operations, while the orthogonal upper decomposition requires Np^2 operations. Depending on the relative size of N and p, the Cholesky can sometimes be faster; on the other hand, it can be less numerically stable. Computation of the L1 Lasso via the least angle regression algorithm has the same order of computation as a least squares fit.

4 Linear Methods For Classification

4.1 Introduction

In this chapter we revisit the classification problem and focus on linear methods for classification. Since our predictor $G(x)$ takes values in a discrete set G, we can always divide the input space into a collection of regions labeled according to the classification. We saw in Chapter 2 that the boundaries of these regions can be rough or smooth, depending on the prediction function. For an important class of procedures, these decision boundaries are linear; this is what we will mean by linear methods for classification.

The input space is divided into regions of constant classification, with piecewise hyperplanar decision boundaries. This regression approach is a member of a class of methods that model discriminant functions $\delta_k(x)$ for each class, and then classify x to the class with the largest value for its discriminant function. Strictly speaking, a hyperplane passes through the origin, while an affine set need not. We sometimes ignore the distinction and refer in general to hyperplanes. that model the posterior probabilities $\Pr(G = k|X = x)$ are also in this class. Clearly, if either the $\delta_k(x)$ or $\Pr(G = k|X = x)$ are linear in x, then the decision boundaries will be linear.

4.2 Linear Regression Of An Indicator Matrix

The N training instances of these form an $N \times K$ indicator response matrix Y. Y is a matrix of 0s and 1s, with each row having a single 1 e.g. being a dummy variable. We fit a linear regression model to each of the columns of Y simultaneously.

Nice image of a linear discriminant analysis producing 2 linear boundaries separating 3 classes.

4.3 Linear Discriminant Analysis

Nice image of 3 Gaussian distributions with the same covariance and different means. And the Bayes decision boundaries between each pair of classes are shown.

Quadratic Discriminant Analysis

Nice image has a caption: 2 methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using linear discriminant analysis in the 5-dimensional space $X_1, X_2, X_1X_2, X_1^2, X_2^2$). The right plot shows the quadratic decision boundaries found by quadratic discriminant analysis. The differences are small, as is usually the case. Now I did not think that the differences were so small.

Regularised Discriminant Analysis. A compromise between linear discriminant analysis and quadratic discriminant analysis, which allows one to shrink the

separate covariances of quadratic discriminant analysis toward a common covariance as in linear discriminant analysis. These methods are very similar in flavour to L2 Ridge regression.

Reduced-Rank Linear Discriminant Analysis. The K centroids in p-dimensional input space lie in an affine subspace of dimension $\leq K-1$, and if p is much larger than K, this will be a considerable drop in dimension. Moreover, in locating the closest centroid, we can ignore distances orthogonal to this subspace, since they will contribute equally to each class. Thus we might just as well project the X^* onto this centroid-spanning subspace H_{K-1}, and make distance comparisons there. Thus there is a fundamental dimension reduction in linear discriminant analysis, namely, that we need only consider the data in a subspace of dimension at most $K-1$. If $K=3$, for instance, this could allow us to view the data in a twodimensional plot, color-coding the classes. In doing so we would not have relinquished any of the information needed for linear discriminant analysis classification. What if $K>3$? We might then ask for a $L<K-1$ dimensional subspace $H_L \subseteq H_{K-1}$ optimal for linear discriminant analysis in some sense. Fisher defined optimal to mean that the projected centroids were spread out as much as possible in terms of variance. This amounts to finding principal component subspaces of the centroids themselves.

Gaussian Classification.

Nice image caption: Decision boundaries for the vowel training data, in the two-dimensional subspace spanned by the first two canonical variates. Note that in any higher-dimensional subspace, the decision boundaries are higher-dimensional affine planes, and could not be represented as lines. Looks kind of like that Voronoi diagram of course.

4.4 Logistic Regression

The logistic regression model arises from the desire to model the posterior probabilities of the K classes via linear functions in x, while at the same time ensuring that they sum to one and remain in $[0,1]$. The model is specified in terms of $K-1$ log-odds or logit transformations (reflecting the constraint that the probabilities sum to one). Although the model uses the last class as the denominator in the odds-ratios, the choice of denominator is arbitrary in that the estimates are equivariant under this choice.

Fitting Logistic Regression Models. Logistic regression models are usually fit by maximum likelihood, using the conditional likelihood of G given X. Since $\Pr(G|X)$ completely specifies the

conditional distribution, the multinomial distribution is appropriate.

For the multiclass case ($K \geq 3$) the Newton algorithm can also be expressed as an iteratively reweighted least squares algorithm, but with a vector of $K-1$ responses and a nondiagonal weight matrix per observation. The latter precludes any simplified algorithms, and in this case it is numerically more convenient to work with the expanded vector θ directly. Alternatively coordinate-descent methods can be used to maximise the log-likelihood efficiently. The R package glmnet can fit very large logistic regression problems efficiently, both in N and p. Although designed to fit regularised models, options allow for unregularised fits. Logistic regression models are used mostly as a data analysis and inference tool, where the goal is to understand the role of the input variables in explaining the outcome. Typically many models are fit in a search for a parsimonious model involving a subset of the variables, possibly with some interactions terms.

Quadratic Approximations And Inference. L_1 Regularised Logistic Regression.

4.5 Separating Hyperplanes

We have seen that linear discriminant analysis and logistic regression both estimate linear decision boundaries in similar but slightly different ways. For the rest of this chapter we describe separating hyperplane classifiers. These procedures construct linear decision boundaries that explicitly try to separate the data into different classes as well as possible. They provide the basis for support vector classifiers.

Perceptron Learning Algorithm. The perceptron learning algorithm tries to find a separating hyperplane by minimising the distance of misclassified points to the decision boundary. The algorithm in fact uses stochastic gradient descent to minimise this piecewise linear criterion. This means that rather than computing the sum of the gradient contributions of each observation followed by a step in the negative gradient direction, a step is taken after each observation is visited.

Optimal Separating Hyperplanes. The optimal separating hyperplane separates the two classes and maximises the distance to the closest point from either class. Not only does this provide a unique solution to the separating hyperplane problem, but by maximising the margin between the two classes on the training data, this leads to better classification performance on test data.

5 Basis Expansions And Regularisation

5.1 Introduction

In this chapter and the next we discuss popular methods for moving beyond linearity. The core idea in this chapter

is to augment/replace the vector of inputs X with additional variables, which are transformations of X, and then use linear models in this new space of derived input features.

5.2 Piecewise Polynomials And Splines

More generally, an order-M spline with knots $\epsilon_j, j = 1, \ldots, K$ is a piecewise-polynomial of order M, and has continuous derivatives up to order $M - 2$. A cubic spline has $M = 4$. There is seldom any good reason to go beyond cubic-splines, unless one is interested in smooth derivatives. In practice the most widely used orders are $M = 1$,2 and 4. These fixed-knot splines are also known as regression splines. One needs to select the order of the spline, the number of knots and their placement. One simple approach is to parameterise a family of splines by the number of basis functions or degrees of freedom, and have the observations x_i determine the positions of the knots.

Natural Cubic Splines.

5.3 Filtering And Feature Extraction

In the previous example, we constructed a $p \times M$ basis matrix H, and then transformed our features x into new features $x^* = H^T x$. These filtered versions of the features were then used as inputs into a learning procedure: in the previous example, this was linear logistic regression. Preprocessing of high-dimensional features is a very general and powerful method for improving the performance of a learning algorithm. The preprocessing need not be linear as it was above, but can be a general (nonlinear) function of the form $x^* = g(x)$. The derived features x^* can then be used as inputs into any (linear or nonlinear) learning procedure.

5.4 Smoothing Splines

Here we discuss a spline basis method that avoids the knot selection problem completely by using a maximal set of knots. The complexity of the fit is controlled by regularisation. Consider the following problem: among all functions $f(x)$ with two continuous derivatives, find one that minimises the penalised residual sum of squares where λ is a fixed smoothing parameter. The first term measures closeness to the data, while the second term penalises curvature in the function, and λ establishes a tradeoff between the two.

Degrees Of Freedom And Smoother Matrices. We have not yet indicated how λ is chosen for the smoothing spline. Later in this chapter we describe automatic methods using techniques such as cross-validation. In this section we discuss intuitive ways of prespecifying the amount of smoothing.

5.5 Automatic Selection Of The Smoothing Parameters

Selecting the placement and number of knots for regression splines can be a

combinatorially complex task, unless some simplifications are enforced. The Multivariate Adaptive Regression Splines procedure in Chapter 9 uses a greedy algorithm with some additional approximations to achieve a practical compromise.

Fixing The Degrees Of Freedom. The Bias-Variance Tradeoff.

5.6 Nonparametric Logistic Regression

The smoothing spline problem (5.9) in Section 5.4 is posed in a regression setting. It is typically straightforward to transfer this technology to other domains. Here we consider logistic regression with a single quantitative input X. Fitting $f(x)$ in a smooth fashion leads to a smooth estimate of the conditional probability $\Pr(Y = 1|x)$, which can be used for classification or risk scoring.

5.7 Multidimensional Splines

So far we have focused on one-dimensional spline models. Each of the approaches have multidimensional analogs. Suppose $X \in \mathbb{R}^2$, and we have a basis of functions $h_{1k}(X_1), k = 1, \ldots, M_1$ for representing functions of coordinate X_1, and likewise a set of M_2 functions $h_{2k}(X_2)$ for coordinate X_2. Then the $M_1 \times M_2$ dimensional tensor product basis defined by $g_{jk}(X) = h_{1j}(X_1)h_{2k}(X_2), j = 1, \ldots, M_1, k = 1, \ldots, M_2$ can be used for representing a 2-dimensional function.

5.8 Regularisation And Reproducing Kernel Hilbert Spaces

Spaces of Functions Generated by Kernels. An important subclass of problems of the form are generated by a positive definite kernel $K(x, y)$, and the corresponding space of functions H_K is called a reproducing kernel Hilbert space. The penalty functional J is defined in terms of the kernel as well.

There is a Bayesian interpretation of this class of models, in which f is interpreted as a realisation of a zero-mean stationary Gaussian process, with prior covariance function K. The eigen-decomposition produces a series of orthogonal eigen-functions $\phi_j(x)$ with associated variances γ_j. The typical scenario is that "smooth" functions ϕ_j have large prior variance, while "rough" ϕ_j have small prior variances. The penalty in is the contribution of the prior to the joint likelihood, and penalises more those components with smaller prior variance.

Penalised Polynomial Regression. Gaussian Radial Basis Functions. Support Vector Classifiers.

5.9 Wavelet Smoothing

We have seen two different modes of operation with dictionaries of basis functions. With regression splines, we select a subset of the bases, using either

subject-matter knowledge, or else automatically. The more adaptive procedures such as Multivariate Adaptive Regression Splines can capture both smooth and nonsmooth behavior. With smoothing splines, we use a complete basis, but then shrink the coefficients toward smoothness.

Wavelet Bases And The Wavelet Transform. Adaptive Wavelet Filtering.

6 Kernel Smoothing Methods

6.1 One-Dimensional Kernel Smoothers

This discontinuity is ugly and unnecessary. Rather than give all the points in the neighborhood equal weight, we can assign weights that die off smoothly with distance from the target point. The right panel shows an example of this, using the so-called Nadaraya - Watson kernel-weighted average with the Epanechnikov quadratic kernel.

Local Linear Regression. We have progressed from the raw moving average to a smoothly varying locally weighted average by using kernel weighting. The smooth kernel fit still has problems, however, as exhibited in Figure 6.3 (left panel). Locally weighted averages can be badly biased on the boundaries of the domain because of the asymmetry of the kernel in that region. By fitting straight lines rather than constants locally, we can remove this bias exactly to first order.

Local Polynomial Regression. Why stop at local linear fits? We can fit local polynomial fits of any degree d. In fact, an expansion such as will tell us that the bias will only have components of degree $d+1$ and higher. Figure 6.5 illustrates local quadratic regression. Local linear fits tend to be biased in regions of curvature of the true function, a phenomenon referred to as trimming the hills and filling the valleys. Local quadratic regression is generally able to correct this bias. There is of course a price to be paid for this bias reduction, and that is increased variance.

6.2 Selecting The Width Of The Kernel

There is a natural bias-variance tradeoff as we change the width of the averaging window, which is most explicit for local averages: If the window is narrow, $\hat{f}(x_0)$ is an average of a small number of y_i close to x_0, and its variance will be relatively large-close to that of an individual y_i. The bias will tend to be small. If the window is wide, the variance of $\hat{f}(x_0)$ will be small relative to the variance of any y_i, because of the effects of averaging. The bias will be higher.

6.3 Local Regression In $\mathbb{R}^p$

Kernel smoothing and local regression generalise very naturally to two or more dimensions. The Nadaraya-Watson kernel smoother fits a constant locally with weights supplied by a

p-dimensional kernel. Local linear regression will fit a hyperplane locally in X, by weighted least squares, with weights supplied by a p-dimensional kernel. It is simple to implement and is generally preferred to the local constant fit for its superior performance on the boundaries.

While boundary effects are a problem in one-dimensional smoothing, they are a much bigger problem in two or higher dimensions, since the fraction of points on the boundary is larger. In fact, one of the manifestations of the curse of dimensionality is that the fraction of points close to the boundary increases to one as the dimension grows. Directly modifying the kernel to accommodate two-dimensional boundaries becomes very messy, especially for irregular boundaries. Local polynomial regression seamlessly performs boundary correction to the desired order in any dimensions.

Local regression becomes less useful in dimensions much higher than two or three. We have discussed in some detail the problems of dimensionality, for example. It is impossible to simultaneously maintain localness (low bias) and a sizable sample in the neighborhood (low variance) as the dimension increases, without the total sample size increasing exponentially in p.

6.4 Structured Local Regression Models In $\mathbb{R}^p$

When the dimension to sample-size ratio is unfavorable, local regression does not help us much, unless we are willing to make some structural assumptions about the model. Much of this book is about structured regression and classification models. Here we focus on some approaches directly related to kernel methods.

Structured Kernels. Structured Regression Functions.

6.5 Local Likelihood And Other Models

The concept of local regression and varying coefficient models is extremely broad: any parametric model can be made local if the fitting method accommodates observation weights.

6.6 Kernel Density Estimation And Classification

Kernel density estimation is an unsupervised learning procedure, which historically precedes kernel regression. It also leads naturally to a simple family of procedures for nonparametric classification.

Kernel Density Estimation. Kernel Density Classification. The Naive Bayes Classifier.

6.7 Radial Basis Functions And Kernels

The art of flexible modeling using basis expansions consists of picking an appropriate family of basis functions,

and then controlling the complexity of the representation by selection, regularisation, or both. Some of the families of basis functions have elements that are defined locally; for example, B-splines are defined locally in $\mathbb{R}$. If more flexibility is desired in a particular region, then that region needs to be represented by more basis functions (which in the case of B-splines translates to more knots). Tensor products of $\mathbb{R}$-local basis functions deliver basis functions local in $\mathbb{R}^p$. Not all basis functions are local-for example, the truncated power bases for splines, or the sigmoidal basis functions used in neural-networks (see Chapter 11). The composed function $f(x)$ can nevertheless show local behavior, because of the particular signs and values of the coefficients causing cancellations of global effects. For example, the truncated power basis has an equivalent B-spline basis for the same space of functions; the cancellation is exact in this case. Kernel methods achieve flexibility by fitting simple models in a region local to the target point x_0. Localisation is achieved via a weighting kernel K_λ, and individual observations receive weights $K_\lambda(x_0, x_i)$. Radial basis functions combine these ideas, by treating the kernel functions $K_\lambda(\epsilon, x)$ as basis functions.

6.8 Mixture Models For Density Estimation And Classification

The mixture model is a useful tool for density estimation, and can be viewed as a kind of kernel method. The Gaussian mixture model has the form $f(x) = \sum_{m=1}^{M} \alpha_m \phi(x; \mu_m, \Sigma_m)$ with mixing proportions α_m, $\sum_m \alpha_m = 1$, and each Gaussian density has a mean μ_m and covariance matrix Σ_m. In general, mixture models can use any component densities in place of the Gaussian in (6.32): the Gaussian mixture model is by far the most popular. The parameters are usually fit by maximum likelihood, using the Expectation Maximisation algorithm as described in Chapter 8.

6.9 Computational Considerations

Kernel and local regression and density estimation are memory-based methods: the model is the entire training data set, and the fitting is done at evaluation or prediction time. For many real-time applications, this can make this class of methods infeasible.

The computational cost to fit at a single observation x_0 is $O(N)$ flops, except in oversimplified cases (such as square kernels). By comparison, an expansion in M basis functions costs $O(M)$ for one evaluation, and typically $M \approx O(\log N)$. Basis function methods have an initial cost of at least $O(NM^2 + M^3)$.

The smoothing parameter(s) λ for kernel methods are typically determined off-line, for example using

cross-validation, at a cost of $O(N^2)$ flops.

Popular implementations of local regression, such as the loess function in S-PLUS and R and the locfit procedure, use triangulation schemes to reduce the computations. They compute the fit exactly at M carefully chosen locations ($O(NM)$), and then use blending techniques to interpolate the fit elsewhere ($O(M)$ per evaluation).

7 Model Assessment And Selection

7.1 Introduction

The generalisation performance of a learning method relates to its prediction capability on independent test data. Assessment of this performance is extremely important in practice, since it guides the choice of learning method or model, and gives us a measure of the quality of the ultimately chosen model.

7.2 Bias, Variance, And Model Complexity

We would like to know the expected test error of our estimated model $\hat{f}$. As the model becomes more and more complex, it uses the training data more and is able to adapt to more complicated underlying structures. Hence there is a decrease in bias but an increase in variance. There is some intermediate model complexity that gives minimum expected test error. Unfortunately training error is not a good estimate of the test error, as seen in Figure 7.1. Training error consistently decreases with model complexity, typically dropping to zero if we increase the model complexity enough. However, a model with zero training error is overfit to the training data and will typically generalise poorly.

Model selection: estimating the performance of different models in order to choose the best one.

Model assessment: having chosen a final model, estimating its prediction error (generalisation error) on new data.

If we are in a data-rich situation, the best approach for both problems is to randomly divide the dataset into three parts: a training set, a validation set, and a test set. The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalisation error of the final chosen model. Ideally, the test set should be kept in a "vault," and be brought out only at the end of the data analysis. Suppose instead that we use the test-set repeatedly, choosing the model with smallest test-set error. Then the test set error of the final chosen model will underestimate the true test error, sometimes substantially. Of course pinging a test set multiple time as guidance during a process is essentially just using the test set as a training set and thus will lead to

overfitting on the test set itself.

It is difficult to give a general rule on how to choose the number of observations in each of the three parts, as this depends on the signal-tonoise ratio in the data and the training sample size. A typical split might be 50 for training, and 25 each for validation and testing.

The methods in this chapter are designed for situations where there is insufficient data to split it into three parts. Again it is too difficult to give a general rule on how much training data is enough; among other things, this depends on the signal-to-noise ratio of the underlying function, and the complexity of the models being fit to the data.

7.3 The Bias-Variance Decomposition

The methods in this chapter are designed for situations where there is insufficient data to split it into three parts. Again it is too difficult to give a general rule on how much training data is enough; among other things, this depends on the signal-to-noise ratio of the underlying function, and the complexity of the models being fit to the data.

7.4 Optimism Of The Training Error Rate

Discussions of error rate estimation can be confusing, because we have to make clear which quantities are fixed and which are random. Generalisation error, expected error, training error, true error, extra-sample error, in-sample error, optimism, average optimism, effective number of parameters.

7.5 Estimates Of In-Sample Prediction Error

Squared error loss, leads to a version of the so-called C_p statistic.

The Akaike Information Criterion is a similar but more generally applicable estimate of $\mathrm{Err}_{\mathrm{in}}$ when a log-likelihood loss function is used.

7.6 The Effective Number Of Parameters

The concept of "number of parameters" can be generalised, especially to models where regularisation is used in the fitting. Suppose we stack the outcomes $y_1, y_2, \ldots, y_N$ into a vector y, and similarly for the predictions $\hat{y}$. Then a linear fitting method is one for which we can write $\hat{y} = Sy$, where S is an $N \times N$ matrix depending on the input vectors xi but not on the y_i. Linear fitting methods include linear regression on the original features or on a derived basis set, and smoothing methods that use quadratic shrinkage, such as L2 Ridge regression and cubic smoothing splines. Then the effective number of parameters is defined as $\mathrm{df}(S) = \mathrm{trace}(S)$, the sum of the diagonal elements of S (also known as

the effective degrees-of-freedom). Note that if S is an orthogonal-projection matrix onto a basis set spanned by M features, then $\text{trace}(S) = M$. It turns out that $\text{trace}(S)$ is exactly the correct quantity to replace d as the number of parameters in the C_p statistic.

If y arises from an additive-error model... for models like neural networks, in which we minimise an error function $R(w)$ with weight decay penalty (regularisation) $\alpha \sum_m w_m^2$, the effective number of parameters has the form...

7.7 The Bayesian Approach And The Bayesian Information Criterion

The Bayesian Information Criterion, like the Akaike Information Criterion, is applicable in settings where the fitting is carried out by maximisation of a log-likelihood.

For model selection purposes, there is no clear choice between Akaike Information Criterion and the Bayesian Information Criterion. The Bayesian Information Criterion is asymptotically consistent as a selection criterion. What this means is that given a family of models, including the true model, the probability that the Bayesian Information Criterion will select the correct model approaches one as the sample size $N \to \infty$. This is not the case for the Akaike Information Criterion, which tends to choose models which are too complex as $N \to \infty$. On the other hand, for finite samples, the Bayesian Information Criterion often chooses models that are too simple, because of its heavy penalty on complexity.

7.8 Minimum Description Length

The minimum description length approach gives a selection criterion formally identical to the Bayesian Information Criterion approach, but is motivated from an optimal coding viewpoint.

7.9 Vapnik-Chervonenkis Dimension

The Vapnik-Chervonenkis Dimension of the class $\{f(x, \alpha)\}$ is defined to be the largest number of points (in some configuration) that can be shattered by members of $\{f(x, \alpha)\}$. A set of points is said to be shattered by a class of functions if, no matter how we assign a binary label to each point, a member of the class can perfectly separate them.

7.10 Cross-Validation

Probably the simplest and most widely used method for estimating prediction error is cross-validation. This method directly estimates the expected extra-sample error
$\text{Err} = E[L(Y, \hat{f}(X))]$, the average generalisation error when the method $\hat{f}(X)$ is applied to an independent test sample from the joint distribution of X and Y. As mentioned earlier, we might hope that cross-validation estimates the

conditional error, with the training set T held fixed. But as we will see in Section 7.12, cross-validation typically estimates well only the expected prediction error.

K-Fold Cross-Validation.

The Wrong and Right Way to Do Cross-validation. 1. Divide the samples into K cross-validation folds (groups) at random. 2. For each fold $k = 1, 2, \ldots, K$ (a) Find a subset of "good" predictors that show fairly strong (univariate) correlation with the class labels, using all of the samples except those in fold k. (b) Using just this subset of predictors, build a multivariate classifier, using all of the samples except those in fold k. (c) Use the classifier to predict the class labels for the samples in fold k.

7.11 Bootstrap Methods

The bootstrap is a general tool for assessing statistical accuracy. First we describe the bootstrap in general, and then show how it can be used to estimate extra-sample prediction error. As with cross-validation, the bootstrap seeks to estimate the conditional error Err_T, but typically estimates well only the expected prediction error Err.

7.12 Conditional Or Expected Test Error

We conclude that estimation of test error for a particular training set is not easy in general, given just the data from that same training set. Instead, cross-validation and related methods may provide reasonable estimates of the expected error Err.

8 Model Inference And Averaging

8.1 Introduction

For most of this book, the fitting (learning) of models has been achieved by minimising a sum of squares for regression, or by minimising cross-entropy for classification. In fact, both of these minimisations are instances of the maximum likelihood approach to fitting. In this chapter we provide a general exposition of the maximum likelihood approach, as well as the Bayesian method for inference. The bootstrap, introduced in Chapter 7, is discussed in this context, and its relation to maximum likelihood and Bayes is described. Finally, we present some related techniques for model averaging and improvement, including committee methods, Bootstrap Aggregating, stacking and bumping.

8.2 The Bootstrap And Maximum Likelihood Methods

The bootstrap method provides a direct computational way of assessing uncertainty, by sampling from the training data.

The bootstrap method described above, in which we sample with replacement

from the training data, is called the nonparametric bootstrap. This really means that the method is "model-free," since it uses the raw data, not a specific parametric model, to generate new datasets. Consider a variation of the bootstrap, called the parametric bootstrap, in which we simulate new responses by adding Gaussian noise to the predicted values.

Maximum Likelihood Inference. It turns out that the parametric bootstrap agrees with least squares in the previous example because the model (8.5) has additive Gaussian errors. In general, the parametric bootstrap agrees not with least squares but with maximum likelihood.

In essence the bootstrap is a computer implementation of nonparametric or parametric maximum likelihood. The advantage of the bootstrap over the maximum likelihood formula is that it allows us to compute maximum likelihood estimates of standard errors and other quantities in settings where no formulas are available.

8.3 Bayesian Methods

In the Bayesian approach to inference, we specify a sampling model $\Pr(Z|\theta)$ (density or probability mass function) for our data given the parameters, and a prior distribution for the parameters $\Pr(\theta)$ reflecting our knowledge about θ before we see the data. We then compute the posterior distribution $\Pr(\theta|Z) = \frac{\Pr(Z|\theta)\cdot\Pr(\theta)}{\int \Pr(Z|\theta)\cdot\Pr(\theta)d\theta}$ which represents our updated knowledge about θ after we see the data. To understand this posterior distribution, one might draw samples from it or summarise by computing its mean or mode. The Bayesian approach differs from the standard ("frequentist") method for inference in its use of a prior distribution to express the uncertainty present before seeing the data, and to allow the uncertainty remaining after seeing the data to be expressed in the form of a posterior distribution.

8.4 Relationship Between In The Bootstrap And Bayesian Inference

Consider first a very simple example, in which we observe a single observation z from a normal distribution $z \approx N(\theta, 1)$. To carry out a Bayesian analysis for θ, we need to specify a prior. The most convenient and common choice would be $\theta \approx N(0, \tau)$ giving posterior distribution $\theta|z \approx N\left(\frac{z}{1+\frac{1}{\tau}}, \frac{1}{1+\frac{1}{\tau}}\right)$. This is the same as a parametric bootstrap distribution in which we generate bootstrap values z^* from the maximum likelihood estimate of the sampling density $N(z, 1)$.

8.5 The Expectation Maximisation Algorithm

The Expectation Maximisation algorithm is a popular tool for simplifying difficult maximum likelihood problems. We first describe it

in the context of a simple mixture model. In the expectation step, we do a soft assignment of each observation to each model: the current estimates of the parameters are used to assign responsibilities according to the relative density of the training points under each model. In the maximisation step, these responsibilities are used in weighted maximum-likelihood fits to update the estimates of the parameters.

Here is a different view of the Expectation Maximisation procedure, as a joint maximisation algorithm.

8.6 Markov Chain Monte Carlo For Sampling From The Posterior

Having defined a Bayesian model, one would like to draw samples from the resulting posterior distribution, in order to make inferences about the parameters. Except for simple models, this is often a difficult computational problem. In this section we discuss the Markov Chain Monte Carlo approach to posterior sampling. We will see that Gibbs sampling, a Markov Chain Monte Carlo procedure, is closely related to the Expectation Maximisation algorithm: the main difference is that it samples from the conditional distributions rather than maximising over them.

8.7 Bootstrap Aggregating

Earlier we introduced the bootstrap as a way of assessing the accuracy of a parameter estimate or a prediction. Here we show how to use the bootstrap to improve the estimate or prediction itself. In Section 8.4 we investigated the relationship between the bootstrap and Bayes approaches, and found that the bootstrap mean is approximately a posterior average. Bootstrap Aggregating further exploits this connection.

Nice image caption: Data with two features and two classes, separated by a linear boundary. (Left panel:) Decision boundary estimated from Bootstrap Aggregating the decision rule from a single split, axis-oriented classifier. (Right panel:) Decision boundary from boosting the decision rule of the same classifier. The test error rates are 0.166, and 0.065, respectively. Boosting is described in Chapter 10.

8.8 Model Averaging And Stacking

In Section 8.4 we viewed bootstrap values of an estimator as approximate posterior values of a corresponding parameter, from a kind of nonparametric Bayesian analysis. Viewed in this way, the bagged estimate (8.51) is an approximate posterior Bayesian mean. In contrast, the training sample estimate $\hat{f}(x)$ corresponds to the mode of the posterior. Since the posterior mean (not mode) minimises squared-error loss, it is not surprising that Bootstrap Aggregating can often reduce mean squared-error. Here we discuss Bayesian

model averaging more generally. We have a set of candidate models $M_m, m = 1, \ldots, M$ for our training set Z. These models may be of the same type with different parameter values (e.g., subsets in linear regression), or different models for the same task (e.g., neural networks and regression trees). Suppose ζ is some quantity of interest, for example, a prediction $f(x)$ at some fixed feature value x. The posterior distribution of ζ is... with posterior mean... This Bayesian prediction is a weighted average of the individual predictions, with weights proportional to the posterior probability of each model. This formulation leads to a number of different model-averaging strategies. Committee methods take a simple unweighted average of the predictions from each model, essentially giving equal probability to each model. More ambitiously, the development in Section 7.7 shows the Bayesian Information Criterion can be used to estimate posterior model probabilities. This is applicable in cases where the different models arise from the same parametric model, with different parameter values. The Bayesian Information Criterion gives weight to each model depending on how well it fits and how many parameters it uses. One can also carry out the Bayesian recipe in full.

8.9 Stochastic Search: Bumping

The final method described in this chapter does not involve averaging or combining models, but rather is a technique for finding a better single model. Bumping uses bootstrap sampling to move randomly through model space. For problems where fitting method finds many local minima, bumping can help the method to avoid getting stuck in poor solutions.

As in Bootstrap Aggregating, we draw bootstrap samples and fit a model to each. But rather than average the predictions, we choose the model estimated from a bootstrap sample that best fits the training data. In detail, we draw bootstrap samples and fit our model to each, giving predictions at input point x. We then choose the model that produces the smallest prediction error, averaged over the original training set. For squared error, for example, we choose the model obtained from bootstrap sample $\hat{b}$. By convention we also include the original training sample in the set of bootstrap samples, so that the method is free to pick the original model if it has the lowest training error. By perturbing the data, bumping tries to move the fitting procedure around to good areas of model space. For example, if a few data points are causing the procedure to find a poor solution, any bootstrap sample that omits those data points should procedure a better solution.

9 Additive Models, Trees, And Related Methods

9.1 Generalised Additive Models

Regression models play an important role in many data analyses, providing prediction and classification rules, and data analytic tools for understanding the importance of different inputs. Although attractively simple, the traditional linear model often fails in these situations: in real life, effects are often not linear. In earlier chapters we described techniques that used predefined basis functions to achieve nonlinearities. This section describes more automatic flexible statistical methods that may be used to identify and characterise nonlinear regression effects. These methods are called "generalised additive models."

9.2 Tree-Based Methods

Tree-based methods partition the feature space into a set of rectangles, and then fit a simple model (like a constant) in each one. They are conceptually simple yet powerful. We first describe a popular method for tree-based regression and classification called Classification And Regression Tree, and later contrast it with C4.5, a major competitor.

Regression Trees. Classification Trees.

Other Issues. Categorical Predictors. The Loss Matrix. Missing Predictor Values. Why Binary Splits? Other Tree-Building Procedures. Linear Combination Splits. Instability Of Trees. Lack Of Smoothness. Difficulty In Capturing Additive Structure.

9.3 Patient Rule Induction Method: Bump Hunting

Tree-based methods (for regression) partition the feature space into boxshaped regions, to try to make the response averages in each box as different as possible. The splitting rules defining the boxes are related to each through a binary tree, facilitating their interpretation. The patient rule induction method also finds boxes in the feature space, but seeks boxes in which the response average is high. Hence it looks for maxima in the target function, an exercise known as bump hunting. (If minima rather than maxima are desired, one simply works with the negative response values.) Patient Rule Induction Method also differs from tree-based partitioning methods in that the box definitions are not described by a binary tree. This makes interpretation of the collection of rules more difficult; however, by removing the binary tree constraint, the individual rules are often simpler. The main box construction method in Patient Rule Induction Method works from the top down, starting with a box containing all of the data. The box is compressed along one face by a small amount, and the observations then falling outside the box are peeled off. The face chosen for compression is the one resulting in the largest box mean,

after the compression is performed. Then the process is repeated, stopping when the current box contains some minimum number of data points.

9.4 Multivariate Adaptive Regression Splines

Multivariate Adaptive Regression Splines is an adaptive procedure for regression, and is well suited for highdimensional problems (i.e., a large number of inputs). It can be viewed as a generalisation of stepwise linear regression or a modification of the Classification And Regression Tree method to improve the latter's performance in the regression setting. We introduce Multivariate Adaptive Regression Splines from the first point of view, and later make the connection to Classification And Regression Tree.

9.5 Hierarchical Mixtures Of Experts

The Hierarchical Mixtures Of Expert procedure can be viewed as a variant of tree-based methods. The main difference is that the tree splits are not hard decisions but rather soft probabilistic ones. At each node an observation goes left or right with probabilities depending on its input values. This has some computational advantages since the resulting parameter optimisation problem is smooth, unlike the discrete split point search in the tree-based approach. The soft splits might also help in prediction accuracy and provide a useful alternative description of the data. There are other differences between Hierarchical Mixtures Of Experts and the Classification And Regression Tree implementation of trees. In an Hierarchical Mixtures Of Expert, a linear (or logistic regression) model is fit in each terminal node, instead of a constant as in Classification And Regression Tree. The splits can be multiway, not just binary, and the splits are probabilistic functions of a linear combination of inputs, rather than a single input as in the standard use of Classification And Regression Tree. However, the relative merits of these choices are not clear, and most were discussed at the end of Section 9.2.

9.6 Missing Data

Suspicious data set consider ignoring feature entirely so as to mitigate against downside risks. Understand there may be settings where imputations confers additional predictive power.

It is quite common to have observations with missing values for one or more input features. The usual approach is to impute (fill-in) the missing values in some way. However, the first issue in dealing with the problem is determining whether the missing data mechanism has distorted the observed data. Roughly speaking, data are missing at random if the mechanism resulting in its omission is independent of its (unobserved) value. A more precise

definition is given in Little and Rubin.

Assuming the features are missing completely at random, there are a number of ways of proceeding: 1. Discard observations with any missing values. 2. Rely on the learning algorithm to deal with missing values in its training phase. 3. Impute all missing values before training.

For most learning methods, the imputation approach is necessary. The simplest tactic is to impute the missing value with the mean or median of the nonmissing values for that feature. (Note that the above procedure for generalised additive models is analogous to this.) If the features have at least some moderate degree of dependence, one can do better by estimating a predictive model for each feature given the other features and then imputing each missing value by its prediction from the model.

9.7 Computational Considerations

10 Boosting And Additive Trees

10.1 Boosting Methods

Boosting is one of the most powerful learning ideas introduced in the last twenty years. It was originally designed for classification problems, but as will be seen in this chapter, it can profitably be extended to regression as well. The motivation for boosting was a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee." From this perspective boosting bears a resemblance to Bootstrap Aggregating and other committee-based approaches. However we shall see that the connection is at best superficial and that boosting is fundamentally different.

10.2 Boosting Fits An Additive Model

The success of boosting is really not very mysterious. The key lies in expression. Boosting is a way of fitting an additive expansion in a set of elementary "basis" functions. Here the basis functions are the individual classifiers $G_m(x) \in \{-1, 1\}$. More generally, basis function expansions take the form $f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$, where $\beta_m, m = 1, 2, \ldots, M$ are the expansion coefficients, and $b(x; \gamma) \in \mathbb{R}$ are usually simple functions of the multivariate argument x, characterised by a set of parameters γ. We discuss basis expansions in some detail in Chapter 5. Additive expansions like this are at the heart of many of the learning techniques covered in this book.

10.3 Forward Stagewise Additive Modeling

Forward stagewise modeling approximates the solution to (10.4) by sequentially adding new basis functions to the expansion without adjusting the parameters and coefficients of those that have already been added. This is outlined in Algorithm 10.2. At each

iteration m, one solves for the optimal basis function $b(x;\gamma_m)$ and corresponding coefficient β_m to add to the current expansion $f_{m-1}(x)$. This produces $f_m(x)$, and the process is repeated. Previously added terms are not modified. For squared-error loss... however, as we show near the end of the next section, squared-error loss is generally not a good choice for classification; hence the need to consider other loss criteria.

10.4 Exponential Loss And Adaptive Boosting

The Adaptive Boosting.M1 algorithm was originally motivated from a very different perspective than presented in the previous section. Its equivalence to forward stagewise additive modeling based on exponential loss was only discovered five years after its inception. By studying the properties of the exponential loss criterion, one can gain insight into the procedure and discover ways it might be improved. The principal attraction of exponential loss in the context of additive modeling is computational; it leads to the simple modular reweighting Adaptive Boosting algorithm. However, it is of interest to inquire about its statistical properties. What does it estimate and how well is it being estimated? The first question is answered by seeking its population minimiser.

10.5 Why Exponential Loss?

The Adaptive Boosting.M1 algorithm was originally motivated from a very different perspective than presented in the previous section. Its equivalence to forward stagewise additive modeling based on exponential loss was only discovered five years after its inception. By studying the properties of the exponential loss criterion, one can gain insight into the procedure and discover ways it might be improved. The principal attraction of exponential loss in the context of additive modeling is computational; it leads to the simple modular reweighting Adaptive Boosting algorithm. However, it is of interest to inquire about its statistical properties. What does it estimate and how well is it being estimated? The first question is answered by seeking its population minimiser.

10.6 Loss Functions And Robustness

In this section we examine the different loss functions for classification and regression more closely, and characterise them in terms of their robustness to extreme data.

10.7 "Off-The-Shelf" Procedures For Data Mining

Predictive learning is an important aspect of data mining. As can be seen from this book, a wide variety of methods have been developed for predictive learning from data. For each particular method there are situations for which it is particularly well suited,

and others where it performs badly compared to the best that can be done with that data. We have attempted to characterise appropriate situations in our discussions of each of the respective methods. However, it is seldom known in advance which procedure will perform best or even well for any given problem. Table 10.1 summarises some of the characteristics of a number of learning methods.

10.8 Example: Spam Data

10.9 Boosting Trees

10.10 Numerical Optimisation Via Gradient Boosting

These algorithms are certainly of note I mean for sure it's one of the most critical central introductory randomised algorithms the notion that of course rather than searching say an entire discrete locus search space at the step size one can rapidly speed up the computation of this step by only searching a small portion and that the best option from this portion iteratedly outperforms descent computationally.

Steepest Descent. Gradient Boosting.

10.11 Right-Sized Trees For Boosting

Historically, boosting was considered to be a technique for combining models, here trees. As such, the tree building algorithm was regarded as a primitive that produced models to be combined by the boosting procedure. In this scenario, the optimal size of each tree is estimated separately in the usual manner when it is built (Section 9.2). A very large (oversized) tree is first induced, and then a bottom-up procedure is employed to prune it to the estimated optimal number of terminal nodes. This approach assumes implicitly that each tree is the last one in the expansion (10.28). Except perhaps for the very last tree, this is clearly a very poor assumption. The result is that trees tend to be much too large, especially during the early iterations. This substantially degrades performance and increases computation. The simplest strategy for avoiding this problem is to restrict all trees to be the same size, $J_m = J \forall m$. At each iteration a J-terminal node regression tree is induced. Thus J becomes a meta-parameter of the entire boosting procedure, to be adjusted to maximise estimated performance for the data at hand.

10.12 Regularisation

Shrinkage. Subsampling.

10.13 Interpretation

Single decision trees are highly interpretable. The entire model can be completely represented by a simple two-dimensional graphic (binary tree) that is easily visualised. Linear combinations of trees (10.28) lose this important feature, and must therefore be interpreted in a different way.

Relative Importance of Predictor Variables. Partial Dependence Plots.

10.14 Illustrations

11 Neural Networks

11.1 Introduction

In this chapter we describe a class of learning methods that was developed separately in different fields-statistics and artificial intelligence-based on essentially identical models. The central idea is to extract linear combinations of the inputs as derived features, and then model the target as a nonlinear function of these features. The result is a powerful learning method, with widespread applications in many fields. We first discuss the projection pursuit model, which evolved in the domain of semiparametric statistics and smoothing. The rest of the chapter is devoted to neural network models.

11.2 Projection Pursuit Regression

11.3 Neural Networks

The term neural network has evolved to encompass a large class of models and learning methods. Here we describe the most widely used "vanilla" neural net, sometimes called the single hidden layer back-propagation network, or single layer perceptron. There has been a great deal of hype surrounding neural networks, making them seem magical and mysterious. As we make clear in this section, they are just nonlinear statistical models, much like the projection pursuit regression model discussed above. A neural network is a two-stage regression or classification model, typically represented by a network diagram as in Figure 11.2. This network applies both to regression or classification. For regression, typically $K = 1$ and there is only one output unit Y_1 at the top. However, these networks can handle multiple quantitative responses in a seamless fashion, so we will deal with the general case. For K-class classification, there are K units at the top, with the kth unit modeling the probability of class k. There are K target measurements $Y_k, k = 1, \ldots, K$, each being coded as a 0 - 1 variable for the kth class.

11.4 Fitting Neural Networks

The neural network model has unknown parameters, often called weights, and we seek values for them that make the model fit the training data well.

11.5 Some Issues In Training Neural Networks

There is quite an art in training neural networks. The model is generally overparametrised, and the optimisation problem is nonconvex and unstable unless certain guidelines are followed. In this section we summarise some of the important issues.

Starting Values. Overfitting. Scaling Of The Inputs. Number Of Hidden Units

And Layers. Multiple Minima.

11.6 Example: Simulated Data

11.7 Example: ZIP Code Data

11.8 Discussion

Both projection pursuit regression and neural networks take nonlinear functions of linear combinations ("derived features") of the inputs. This is a powerful and very general approach for regression and classification, and has been shown to compete well with the best learning methods on many problems. These tools are especially effective in problems with a high signal-to-noise ratio and settings where prediction without interpretation is the goal. They are less effective for problems where the goal is to describe the physical process that generated the data and the roles of individual inputs. Each input enters into the model in many places, in a nonlinear fashion. Some authors (Hinton, 1989) plot a diagram of the estimated weights into each hidden unit, to try to understand the feature that each unit is extracting. This is limited however by the lack of identifiability of the parameter vectors.

11.9 Bayesian Neural Nets And The NIPS 2003 Challenge

11.10 Computational Considerations

With N observations, p predictors, M hidden units and L training epochs, a neural network fit typically requires $O(NpML)$ operations. There are many packages available for fitting neural networks, probably many more than exist for mainstream statistical methods. Because the available software varies widely in quality, and the learning problem for neural networks is sensitive to issues such as input scaling, such software should be carefully chosen and tested.

12 Support Vector Machines And Flexible Discriminants

12.1 Introduction

In this chapter we describe generalisations of linear decision boundaries for classification. Optimal separating hyperplanes are introduced in Chapter 4 for the case when two classes are linearly separable. Here we cover extensions to the nonseparable case, where the classes overlap. These techniques are then generalised to what is known as the support vector machine, which produces nonlinear boundaries by constructing a linear boundary in a large, transformed version of the feature space. The second set of methods generalise Fisher's linear discriminant analysis. The generalisations include flexible discriminant analysis which facilitates construction of nonlinear boundaries in a manner very similar to the support vector machines, penalised discriminant analysis for problems such as signal and image classification where the large number of features are highly

correlated, and mixture discriminant analysis for irregularly shaped classes.

12.2 The Support Vector Classifier

In Chapter 4 we discussed a technique for constructing an optimal separating hyperplane between two perfectly separated classes. We review this and generalise to the nonseparable case, where the classes may not be separable by a linear boundary.

12.3 Support Vector Machines And Kernels

The support vector classifier described so far finds linear boundaries in the input feature space. As with other linear methods, we can make the procedure more flexible by enlarging the feature space using basis expansions such as polynomials or splines (Chapter 5). Generally linear boundaries in the enlarged space achieve better training-class separation, and translate to nonlinear boundaries in the original space.

12.4 Generalising Linear Discriminant Analysis

In Section 4.3 we discussed linear discriminant analysis, a fundamental tool for classification. For the remainder of this chapter we discuss a class of techniques that produce better classifiers than linear discriminant analysis by directly generalising linear discriminant analysis.

12.5 Flexible Discriminant Analysis

In this section we describe a method for performing linear discriminant analysis using linear regression on derived responses. This in turn leads to nonparametric and flexible alternatives to linear discriminant analysis.

12.6 Penalised Discriminant Analysis

Although Flexible Discriminant Analysis is motivated by generalising optimal scoring, it can also be viewed directly as a form of regularised discriminant analysis. Suppose the regression procedure used in Flexible Discriminant Analysis amounts to a linear regression onto a basis expansion $h(X)$, with a quadratic penalty on the coefficients.

12.7 Mixture Discriminant Analysis

Linear discriminant analysis can be viewed as a prototype classifier. Each class is represented by its centroid, and we classify to the closest using an appropriate metric. In many situations a single prototype is not sufficient to represent inhomogeneous classes, and mixture models are more appropriate. In this section we review Gaussian mixture models and show how they can be generalised via the Flexible Discriminant Analysis and Penalised Discriminant Analysis methods discussed earlier. A Gaussian mixture model for the kth class has density... where the mixing proportions π_{kr} sum

to one. This has R_k prototypes for the kth class, and in our specification, the same covariance matrix Σ is used as the metric throughout. Given such a model for each class, the class posterior probabilities are given by where Π_k represent the class prior probabilities.

13 Prototype Methods And Nearest-Neighbours

13.1 Introduction

In this chapter we discuss some simple and essentially model-free methods for classification and pattern recognition. Because they are highly unstructured, they typically are not useful for understanding the nature of the relationship between the features and class outcome. However, as black box prediction engines, they can be very effective, and are often among the best performers in real data problems. The nearest-neighbor technique can also be used in regression; this was touched on in Chapter 2 and works reasonably well for low-dimensional problems. However, with high-dimensional features, the bias-variance tradeoff does not work as favorably for nearestneighbor regression as it does for classification.

13.2 Prototype Methods

Throughout this chapter, our training data consists of the N pairs $(x_1, g_1), \ldots, (x_n, g_N)$ where g_i is a class label taking values in $1, 2, \ldots, K$. Prototype methods represent the training data by a set of points in feature space. These prototypes are typically not examples from the training sample, except in the case of 1-nearest-neighbor classification discussed later.

13.3 k-Nearest-Neighbour Classifiers

These classifiers are memory-based, and require no model to be fit. Given a query point x_0, we find the k training points $x(r), r = 1, \ldots, k$ closest in distance to x_0, and then classify using majority vote among the k neighbors.

13.4 Adaptive Nearest-Neighbour Methods

When nearest-neighbor classification is carried out in a high-dimensional feature space, the nearest neighbors of a point can be very far away, causing bias and degrading the performance of the rule.

What can be done about this problem? Consider the two-class situation in Figure 13.13. There are two features, and a nearest-neighborhood at a query point is depicted by the circular region. Implicit in near-neighbor classification is the assumption that the class probabilities are roughly constant in the neighborhood, and hence simple averages give good estimates. However, in this example the class probabilities vary only in the horizontal direction. If we knew this, we would stretch the neighborhood in the vertical direction,

as shown by the tall rectangular region. This will reduce the bias of our estimate and leave the variance the same.

13.5 Computational Considerations

One drawback of nearest-neighbor rules in general is the computational load, both in finding the neighbors and storing the entire training set. With N observations and p predictors, nearest-neighbor classification requires Np operations to find the neighbors per query point. There are fast algorithms for finding nearest-neighbors which can reduce this load somewhat. Hastie and Simard (1998) reduce the computations for tangent distance by developing analogs of K-means clustering in the context of this invariant metric. Reducing the storage requirements is more difficult, and various editing and condensing procedures have been proposed. The idea is to isolate a subset of the training set that suffices for nearest-neighbor predictions, and throw away the remaining training data. Intuitively, it seems important to keep the training points that are near the decision boundaries and on the correct side of those boundaries, while some points far from the boundaries could be discarded.

14 Unsupervised Learning

14.1 Introduction

In this chapter we address unsupervised learning or "learning without a teacher." In this case one has a set of N observations $(x_1, x_2, \ldots, x_N)$ of a random p-vector X having joint density $\Pr(X)$. The goal is to directly infer the properties of this probability density without the help of a supervisor or teacher providing correct answers or degree-of-error for each observation. The dimension of X is sometimes much higher than in supervised learning, and the properties of interest are often more complicated than simple location estimates. These factors are somewhat mitigated by the fact that X represents all of the variables under consideration; one is not required to infer how the properties of $\Pr(X)$ change, conditioned on the changing values of another set of variables.

14.2 Association Rules

More generally, the basic goal of association rule analysis is to find a collection of prototype X-values $v_1, \ldots, v_L$ for the feature vector X, such that the probability density $\Pr(v_l)$ evaluated at each of those values is relatively large. In this general framework, the problem can be viewed as "mode finding" or "bump hunting." As formulated, this problem is impossibly difficult. A natural estimator for each $\Pr(v_l)$ is the fraction of observations for which $X = v_l$. For problems that involve more than a small number of variables, each of which can assume more than a small number of values, the number of

observations for which $X = v_l$ will nearly always be too small for reliable estimation. In order to have a tractable problem, both the goals of the analysis and the generality of the data to which it is applied must be greatly simplified. The first simplification modifies the goal. Instead of seeking values x where $\Pr(x)$ is large, one seeks regions of the X-space with high probability content relative to their size or support.

14.3 Cluster Analysis

Cluster analysis, also called data segmentation, has a variety of goals. All relate to grouping or segmenting a collection of objects into subsets or "clusters," such that those within each cluster are more closely related to one another than objects assigned to different clusters. An object can be described by a set of measurements, or by its relation to other objects. In addition, the goal is sometimes to arrange the clusters into a natural hierarchy. This involves successively grouping the clusters themselves so that at each level of the hierarchy, clusters within the same group are more similar to each other than those in different groups. Cluster analysis is also used to form descriptive statistics to ascertain whether or not the data consists of a set distinct subgroups, each group representing objects with substantially different properties. This latter goal requires an assessment of the degree of difference between the objects assigned to the respective clusters. Central to all of the goals of cluster analysis is the notion of the degree of similarity (or dissimilarity) between the individual objects being clustered. A clustering method attempts to group the objects based on the definition of similarity supplied to it. This can only come from subject matter considerations. The situation is somewhat similar to the specification of a loss or cost function in prediction problems (supervised learning). There the cost associated with an inaccurate prediction depends on considerations outside the data.

14.4 Self-Organising Maps

This method can be viewed as a constrained version of K-means clustering, in which the prototypes are encouraged to lie in a one- or two-dimensional manifold in the feature space. The resulting manifold is also referred to as a constrained topological map, since the original high-dimensional observations can be mapped down onto the two-dimensional coordinate system. The original SOM algorithm was online-observations are processed one at a time-and later a batch version was proposed. The technique also bears a close relationship to principal curves and surfaces, which are discussed in the next section.

14.5 Principal Components, Curves, And Surfaces

Principal components are discussed in

Sections 3.4.1, where they shed light on the shrinkage mechanism of L2 Ridge regression. Principal components are a sequence of projections of the data, mutually uncorrelated and ordered in variance. In the next section we present principal components as linear manifolds approximating a set of N points $x_i \in \mathbb{R}^p$. We then present some nonlinear generalisations in Section 14.5.2. Other recent proposals for nonlinear approximating manifolds are discussed in Section 14.9.

14.6 Non-Negative Matrix Factorisation

Non-negative matrix factorisation (Lee and Seung, 1999) is a recent alternative approach to principal components analysis, in which the data and components are assumed to be non-negative. It is useful for modeling non-negative data such as images.

14.7 Independent Component Analysis And Exploratory Projection Pursuit

Multivariate data are often viewed as multiple indirect measurements arising from an underlying source, which typically cannot be directly measured.

Factor analysis is a classical technique developed in the statistical literature that aims to identify these latent sources. Factor analysis models are typically wed to Gaussian distributions, which has to some extent hindered their usefulness. More recently, independent component analysis has emerged as a strong competitor to factor analysis, and as we will see, relies on the non-Gaussian nature of the underlying sources for its success.

14.8 Multidimensional Scaling

Both self-organising maps and principal curves and surfaces map data points in $\mathbb{R}^p$ to a lower-dimensional manifold. Multidimensional scaling has a similar goal, but approaches the problem in a somewhat different way.

14.9 Nonlinear Dimension Reduction And Local Multidimensional Scaling

Several methods have been recently proposed for nonlinear dimension reduction, similar in spirit to principal surfaces. The idea is that the data lie close to an intrinsically low-dimensional nonlinear manifold embedded in a high-dimensional space. These methods can be thought of as "flattening" the manifold, and hence reducing the data to a set of low-dimensional coordinates that represent their relative positions in the manifold. They are useful for problems where signal-to-noise ratio is very high (e.g., physical systems), and are probably not as useful for observational data with lower signal-to-noise ratios.

14.10 The Google PageRank Algorithm

To wax historical a little this is possibly one of the most pre eminent seminal of seminal algorithms in the history of

humanity. A mere 23 years ago the state of the world was so different and these dudes came along, composed an algorithm, and boom a couple years later "Google" is part of common English, a firm runs the world, and makes great products like mail, scholar, Chrome, Chrome OS, etc.

In this section we give a brief description of the original PageRank algorithm used by the Google search engine, an interesting recent application of unsupervised learning methods. We suppose that we have N web pages and wish to rank them in terms of importance. For example, the N pages might all contain a string match to "statistical learning" and we might wish to rank the pages in terms of their likely relevance to a websurfer. The PageRank algorithm considers a webpage to be important if many other webpages point to it. However the linking webpages that point to a given page are not treated equally: the algorithm also takes into account both the importance (PageRank) of the linking pages and the number of outgoing links that they have. Linking pages with higher PageRank are given more weight, while pages with more outgoing links are given less weight. These ideas lead to a recursive definition for PageRank, detailed next.

15 Random Forests

15.1 Introduction

Recall this book may not be up to date on the latest and greatest in Kaggle Cython optimised Extreme Gradient Boosting techniques or whatever the latest and greatest smash hit in this domain is I mean quite a few of these can lead to relatively performant models in a private sector setting.

Bootstrap Aggregating or bootstrap aggregation (section 8.7) is a technique for reducing the variance of an estimated prediction function. Bootstrap Aggregating seems to work especially well for high-variance, low-bias procedures, such as trees. For regression, we simply fit the same regression tree many times to bootstrap sampled versions of the training data, and average the result. For classification, a committee of trees each cast a vote for the predicted class. Boosting in Chapter 10 was initially proposed as a committee method as well, although unlike Bootstrap Aggregating, the committee of weak learners evolves over time, and the members cast a weighted vote. Boosting appears to dominate Bootstrap Aggregating on most problems, and became the preferred choice. Random forests (Breiman, 2001) is a substantial modification of Bootstrap Aggregating that builds a large collection of de-correlated trees, and then averages them. On many problems the performance of random forests is very similar to boosting, and

they are simpler to train and tune. As a consequence, random forests are popular, and are implemented in a variety of packages.

15.2 Definition Of Random Forests

The essential idea in Bootstrap Aggregating (Section 8.7) is to average many noisy but approximately unbiased models, and hence reduce the variance. Trees are ideal candidates for Bootstrap Aggregating, since they can capture complex interaction structures in the data, and if grown sufficiently deep, have relatively low bias. Since trees are notoriously noisy, they benefit greatly from the averaging. Moreover, since each tree generated in Bootstrap Aggregating is identically distributed (i.d.), the expectation of an average of B such trees is the same as the expectation of any one of them. This means the bias of bagged trees is the same as that of the individual trees, and the only hope of improvement is through variance reduction. This is in contrast to boosting, where the trees are grown in an adaptive way to remove bias, and hence are not i.d.

15.3 Details Of Random Forests

We have glossed over the distinction between random forests for classification versus regression. When used for classification, a random forest obtains a class vote from each tree, and then classifies using majority vote (see Section 8.7 on Bootstrap Aggregating for a similar discussion). When used for regression, the predictions from each tree at a target point x are simply averaged, as in (15.2). In addition, the inventors make the following recommendations: For classification, the default value for m is $\lfloor\sqrt{p}\rfloor$ and the minimum node size is one. For regression, the default value for m is $\lfloor\frac{p}{3}\rfloor$ and the minimum node size is five. In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters. In Figure 15.3 the $m = 6$ performs much better than the default value $\lfloor\frac{8}{3}\rfloor = 2$.

15.4 Analysis Of Random Forests

In this section we analyze the mechanisms at play with the additional randomisation employed by random forests. For this discussion we focus on regression and squared error loss, since this gets at the main points, and bias and variance are more complex with 0-1 loss (see Section 7.3.1). Furthermore, even in the case of a classification problem, we can consider the random-forest average as an estimate of the class posterior probabilities, for which bias and variance are appropriate descriptors.

16 Ensemble Learning

16.1 Introduction

The idea of ensemble learning is to build a prediction model by combining

the strengths of a collection of simpler base models. We have already seen a number of examples that fall into this category. Bootstrap Aggregating in Section 8.7 and random forests in Chapter 15 are ensemble methods for classification, where a committee of trees each cast a vote for the predicted class. Boosting in Chapter 10 was initially proposed as a committee method as well, although unlike random forests, the committee of weak learners evolves over time, and the members cast a weighted vote. Stacking (Section 8.8) is a novel approach to combining the strengths of a number of fitted models. In fact one could characterise any dictionary method, such as regression splines, as an ensemble method, with the basis functions serving the role of weak learners. Bayesian methods for nonparametric regression can also be viewed as ensemble methods: a large number of candidate models are averaged with respect to the posterior distribution of their parameter settings (e.g. (Neal and Zhang, 2006)). Ensemble learning can be broken down into two tasks: developing a population of base learners from the training data, and then combining them to form the composite predictor. In this chapter we discuss boosting technology that goes a step further; it builds an ensemble model by conducting a regularised and supervised search in a high-dimensional space of weak learners.

16.2 Boosting And Regularisation Paths

In Section 10.12.2 of the first edition of this book, we suggested an analogy between the sequence of models produced by a gradient boosting algorithm and regularised model fitting in high-dimensional feature spaces. This was primarily motivated by observing the close connection between a boosted version of linear regression and the L1 Lasso (Section 3.4.2). These connections have been pursued by us and others, and here we present our current thinking in this area. We start with the original motivation, which fits more naturally in this chapter on ensemble learning.

16.3 Learning Ensembles

The insights learned from the previous sections can be harnessed to produce a more effective and efficient ensemble model. Again we consider functions of the form... where T is a dictionary of basis functions, typically trees. For gradient boosting and random forests, $|T|$ is very large, and it is quite typical for the final model to involve many thousands of trees. In the previous section we argue that gradient boosting with shrinkage fits an L_1 regularised monotone path in this space of trees.

In its simplest form this model could be seen as a way of post-processing boosting or random forests, taking for T_L the collection of trees produced by the gradient boosting or random forest

algorithms. By fitting the L1 Lasso path to these trees, we would typically use a much reduced set, which would save in computations and storage for future predictions. In the next section we describe modifications of this prescription that reduce the correlations in the ensemble T_L, and improve the performance of the L1 Lasso post processor.

17 Undirected Graphical Models

17.1 Introduction

As we will see, the edges in a graph are parametrised by values or potentials that encode the strength of the conditional dependence between the random variables at the corresponding vertices. The main challenges in working with graphical models are model selection (choosing the structure of the graph), estimation of the edge parameters from data, and computation of marginal vertex probabilities and expectations, from their joint distribution. The last two tasks are sometimes called learning and inference in the computer science literature. We do not attempt a comprehensive treatment of this interesting area. Instead, we introduce some basic concepts, and then discuss a few simple methods for estimation of the parameters and structure of undirected graphical models; methods that relate to the techniques already discussed in this book. The estimation approaches that we present for continuous and discrete-valued vertices are different, so we treat them separately. Sections 17.3.1 and 17.3.2 may be of particular interest, as they describe new, regression-based procedures for estimating graphical models.

17.2 Markov Graphs And Their Properties

Many of the methods for estimation and computation on graphs first decompose the graph into its maximal cliques. Relevant quantities are computed in the individual cliques and then accumulated across the entire graph. A prominent example is the join tree or junction tree algorithm for computing marginal and low order probabilities from the joint distribution on a graph. Details can be found in the literature.

17.3 Undirected Graphical Models For Continuous Variables

Here we consider Markov networks where all the variables are continuous. The Gaussian distribution is almost always used for such graphical models, because of its convenient analytical properties. We assume that the observations have a multivariate Gaussian distribution with mean μ and covariance matrix Σ. Since the Gaussian distribution represents at most second-order relationships, it automatically encodes a pairwise Markov graph. The graph in Figure 17.1 is an example of a Gaussian

graphical model. The Gaussian distribution has the property that all conditional distributions are also Gaussian. The inverse covariance matrix $\Sigma - 1$ contains information about the partial covariances between the variables; that is, the covariances between pairs i and j, conditioned on all other variables. In particular, if the ijth component of $\Theta = \Sigma^{-1}$ is zero, then variables i and j are conditionally independent, given the other variables (Exercise 17.3).

17.4 Undirected Graphical Models For Discrete Variables

Undirected Markov networks with all discrete variables are popular, and in particular pairwise Markov networks with binary variables being the most common. They are sometimes called Ising models in the statistical mechanics literature, and Boltzmann machines in the machine learning literature, where the vertices are referred to as "nodes" or "units" and are binary-valued. In addition, the values at each node can be observed ("visible") or unobserved ("hidden"). The nodes are often organised in layers, similar to a neural network. Boltzmann machines are useful both for unsupervised and supervised learning, especially for structured input data such as images, but have been hampered by computational difficulties. Figure 17.6 shows a restricted Boltzmann machine (discussed later), in which some variables are hidden, and only some pairs of nodes are connected. We first consider the simpler case in which all p nodes are visible with edge pairs (j, k) enumerated in E.

18 High-Dimensional Problems $p >> N$

18.1 When p Is Much Bigger Than N

In this chapter we discuss prediction problems in which the number of features p is much larger than the number of observations N, often written $p >> N$. Such problems have become of increasing importance, especially in genomics and other areas of computational biology. We will see that high variance and overfitting are a major concern in this setting. As a result, simple, highly regularised approaches often become the methods of choice. The first part of the chapter focuses on prediction in both the classification and regression settings, while the second part discusses the more basic problem of feature selection and assessment.

18.2 Diagonal Linear Discriminant Analysis

The diagonal linear discriminant analysis classifier is often effective in high dimensional settings. It is also called the "independence rule" in Bickel and Levina (2004), who demonstrate theoretically that it will often outperform standard linear discriminant analysis in high-dimensional problems.

Here the diagonal linear discriminant analysis classifier yielded five misclassification errors for the 20 test samples. One drawback of the diagonal linear discriminant analysis classifier is that it uses all of the features (genes), and hence is not convenient for interpretation. With further regularisation we can do better-both in terms of test error and interpretability. We would like to regularise in a way that automatically drops out features that are not contributing to the class predictions. We can do this by shrinking the classwise mean toward the overall mean, for each feature separately. The result is a regularised version of the nearest centroid classifier, or equivalently a regularised version of the diagonal-covariance form of linear discriminant analysis. We call the procedure nearest shrunken centroids.

18.3 Linear Classifiers With Quadratic Regularisation

Regularised discriminant analysis, regularised multinomial logistic regression, and the support vector machine are more complex methods that try to exploit multivariate information in the data. We describe each in turn, as well as a variety of regularisation methods, including both L_1 and L_2 and some in between.

18.4 Linear Classifiers With L_1 Regularisation

The methods of the previous chapter use an L_2 penalty to regularise their parameters, just as in L2 Ridge regression. All of the estimated coefficients are nonzero, and hence no feature selection is performed. In this section we discuss methods that use L_1 penalties instead, and hence provide automatic feature selection.

18.5 Classification When Features Are Unavailable

In some applications the objects under study are more abstract in nature, and it is not obvious how to define a feature vector. As long as we can fill in an $N \times N$ proximity matrix of similarities between pairs of objects in our database, it turns out we can put to use many of the classifiers in our arsenal by interpreting the proximities as inner-products. Protein structures fall into this category, and we explore an example in Section 18.5.1 below. In other applications, such as document classification, feature vectors are available but can be extremely high-dimensional. Here we may not wish to compute with such high-dimensional data, but rather store the innerproducts between pairs of documents. Often these inner-products can be approximated by sampling techniques. Pairwise distances serve a similar purpose, because they can be turned into centered inner-products. Proximity matrices are discussed in more detail in Chapter 14.

18.6 High-Dimensional Regression: Supervised Principal Components

In this section we describe a simple approach to regression and generalised regression that is especially useful when $p >> N$. We illustrate the method on another microarray data example. The data is taken from Rosenwald et al. (2002) and consists of 240 samples from patients with diffuse large B-cell lymphoma, with gene expression measurements for 7399 genes. The outcome is survival time, either observed or right censored. We randomly divided the lymphoma samples into a training set of size 160 and a test set of size 80. Although supervised principal components is useful for linear regression, its most interesting applications may be in survival studies, which is the focus of this example.

18.7 Feature Assessment And The Multiple-Testing Problem

In the first part of this chapter we discuss prediction models in the $p >> N$ setting. Here we consider the more basic problem of assessing the significance of each of the p features. Consider the protein mass spectrometry example of Section 18.4.1. In that problem, the scientist might not be interested in predicting whether a given patient has prostate cancer. Rather the goal might be to identify proteins whose abundance differs between normal and cancer samples, in order to enhance understanding of the disease and suggest targets for drug development. Thus our goal is to assess the significance of individual features. This assessment is usually done without the use of a multivariate predictive model like those in the first part of this chapter. The feature assessment problem moves our focus from prediction to the traditional statistical topic of multiple hypothesis testing. For the remainder of this chapter we will use M instead of p to denote the number of features, since we will frequently be referring to p-values.